A Parallel Trade Study Architecture for Design Optimization of Complex Systems

Hongman Kim, James Mullins, Scott Ragon, and Grant Soremekun
Phoenix Integration, Inc.

and
Jaroslaw Sobieszczanski-Sobieski
NASA Langley Research Center

Design of a successful product requires evaluating many design alternatives in a limited design cycle time. This can be achieved through leveraging design space exploration tools and available computing resources on the network. This paper presents a parallel trade study architecture to integrate trade study clients and computing resources on a network using Web services. The parallel trade study solution is demonstrated to accelerate design of experiments, genetic algorithm optimization, and a cost as an independent variable (CAIV) study for a space system application.

I. Introduction

Manufacturers are under enormous pressure to deliver superior products earlier than their competitors in the global market. Engineers must reduce their design cycle time to consider more design alternatives in a limited amount of time. Two enabling technologies can play important roles in achieving this goal. First, *design exploration techniques*, such as design of experiments (DOE) studies and multidisciplinary design optimization (MDO) methodologies, search for design alternatives in a systematic way. These techniques help engineers identify promising designs, sometimes non-intuitive ones, and reduce the total computational cost. Second, *load balancing software tools* help to fully utilize all available computing resources: high performance computers, clusters, desktop machines, and computational grid.

Because design studies require evaluating many independent designs, there is ample opportunity to perform those evaluations in parallel. Many common trade study techniques are well suited to the paradigm of coarse grained parallelism; parametric studies, DOE studies, and certain optimization techniques such as genetic algorithms involve the evaluation of independent design cases. Moreover, some MDO techniques deliberately separate coupled systems into subsystems that can be computed concurrently. For example, Bi-Level Integrated System Synthesis (BLISS)[1] enables each disciplinary group to perform series of independent subsystem optimization cases in parallel.

However, it is not simple to incorporate the computing resources as an integral part of the design process. Engineers often have to go off line to perform a series of simulations (e.g., parametric studies) on dedicated clusters, and move the data to a local computer to post-process results. The challenge is to seamlessly integrate trade study tools with computing resources available on the network. Ideally, an engineer should be able to set up a trade study, launch its execution, monitor the execution progress, collect the data, and post-process the data, all without leaving his or her desktop environment. While the evaluation of design alternatives provides opportunities for coarse grained parallelism, the design of complex system adds another challenge. Data dependency of application modules must be maintained when each run case is computed on available computers.

This paper discusses implementing a parallel trade study architecture that effectively combines trade study clients and load balancing tools. A Web-centric architecture is used to manage trade study requests and distribute them to computing resources on the network. The ModelCenter[2] MDO framework is used to create an integrated computational model and to submit independent run cases to the trade study execution Web service. The design scan and optimization tools in ModelCenter utilize the architecture to perform trade studies in parallel. This procedure is demonstrated for design optimization studies of a space system.

II. Parallel Trade Study Support Architecture

This paper presents part of an ongoing project to develop a Web-centric engineering design study environment called CenterLink[3,4]. To take the product design process to the next level, groups of engineers must work together as a part of the design process. The Web centric architecture of CenterLink supports groups of engineers who are participating in design projects. Web technologies can benefit engineering design projects by integrating computing resources, data, and expert knowledge. Standard web technologies such as HTTP servers, Servlet containers, database management systems (DBMS), and Web browsers are mature and ready to be leveraged. The Web server/client model provides wide access to information over the network in a secure manner.

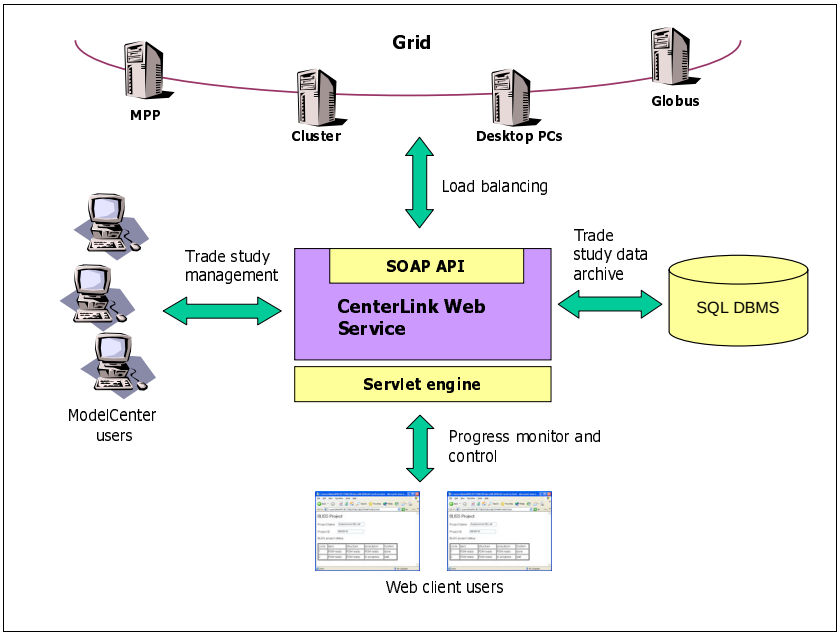

Figure 1: CenterLink Web service integrates distributed computing resources to create an accessible and scalable computing environment.

CenterLink is a set of Web services that integrate computing resources with trade study clients such as ModelCenter[2]. ModelCenter is a process integration and design optimization (PIDO) environment that integrates applications distributed on a network to create complex analysis models. CenterLink accelerates trade studies of ModelCenter by distributing independent run cases of the integrated models over computers on the network. Figure 1 shows a high level architecture of CenterLink. Key features of CenterLink include:

- **Trade Study Management:** ModelCenter users can choose to run trade studies through CenterLink instead of running them on local computer. ModelCenter sends a trade study definition along with an integrated model to CenterLink via CenterLink's API (Application Program Interface) in SOAP (Simple Object Access Protocol) standard. The SOAP API is discussed in more detail in the next section.
- **Load Balancing:** CenterLink redirects the trade study request from ModelCenter to its load balancing service running on a group of PCs or to a cluster that is using a 3rd party load balancing tool, such as LSF[5] (Load Sharing Facility).
- **Trade Study Data Archive:** A standard SQL database is used to store the trade study results in a central location. Large amounts of data can be archived in the database so that the data is readily available for download and post processing in the future.
- **Progress Monitor and Control:** CenterLink allows users to monitor, suspend, restart, and stop trade study processes using Web browsers.

CenterLink works like a switchboard that relays requests from ModelCenter to back-end computing resources. CenterLink serves as the core of the engineering design process by providing individual ModelCenter users with effective ways to share computing resources and data.

III. Web Service Technology

CenterLink leverages Web services technology to create an integrated environment to give engineers the power to solve complex design problems. Traditional Web servers serve HTML pages requested by Web browsers. The Web service technology allows Web servers to host application programs that provide application specific services to client programs over the network. SOAP[6] is de facto standard protocol of Web services based on XML (Extensible Markup Language), an information description language.

Figure 2 describes the Web service model using SOAP. The client program generates service requests written in XML format and sends the request to the Web Server using the HTTP protocol. The Web server parses the XML request and invokes the application program to process the request and returns the results in XML format. Because XML is an extremely versatile data format in plain text, it can easily describe the request and response of sophisticated applications.

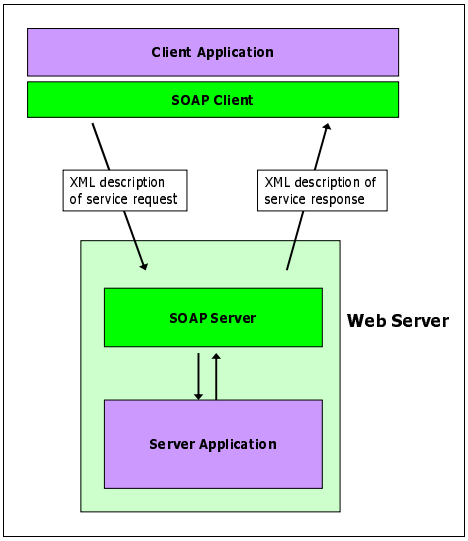

Figure 2: Web service model of SOAP.

The Web services model provides many advantages for implementing the required CenterLink functionalities.

- It provides a powerful mechanism for publishing applications over the network.
- It can create loosely-coupled servers that can talk to each other. Because the connection does not need to be maintained all the time, the SOAP model results in a more robust system.
- It leverages widely accepted Web server technology and protocols providing a scalable and secure service.
- It is based on the XML protocol (independent of any machine specific data format), and can be used on any computer platform.

IV. Computational Load Distribution

Many widely-used trade study techniques are well-suited to the paradigm of coarse-grained parallelism: parametric studies, DOE, and certain optimization techniques (e.g. genetic algorithms), all involve the computation of independent design cases. Moreover, some multidisciplinary design optimization (MDO) techniques deliberately separate coupled systems into subsystems that can be computed concurrently.

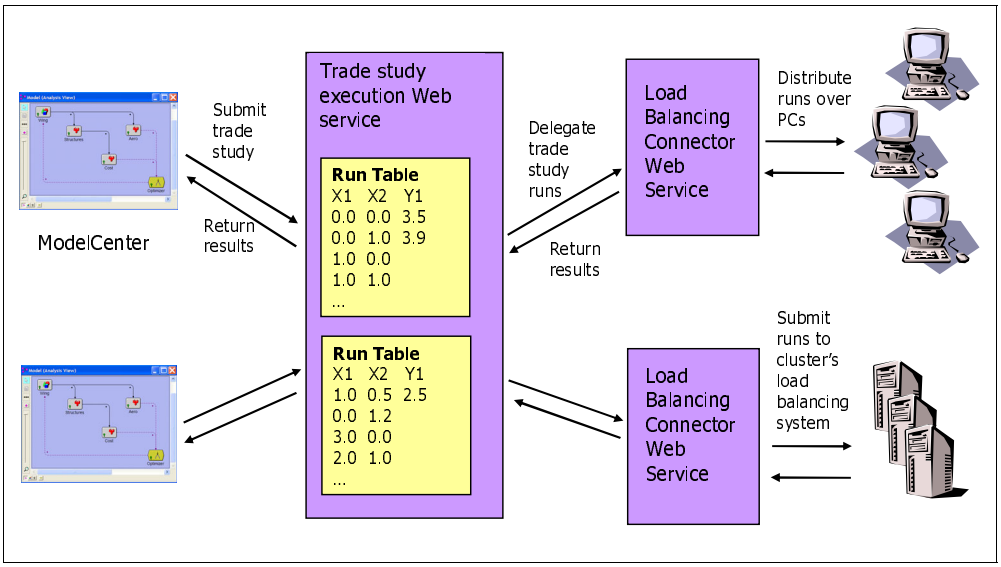

Figure 3: Design studies are accelerated using CenterLink's load balancing service.

Figure 3 illustrates load balancing architecture of CenterLink. Two types of Web services are involved: trade study execution Web services and load balancing connectors. The trade study execution service manages requests from ModelCenter. For example, ModelCenter users submit trade studies to CenterLink and CenterLink keeps a record of the corresponding run tables. The CenterLink Web service delegates execution of the run table to the Load Balancing Connector Web service. Note that the trade study management service and the load balancing connector service may be running on different computers. If the connector is serving a cluster with a load-balancing tool such as LSF, the load balancing connector can utilize the existing load balancing tool. If the connector is serving a group of desktop computers, it will distribute runs over the computers as they become available.

Key features of the load balancing mechanism can be summarized as:

- **Fire and Forget:** Because CenterLink manages the status of runs, ModelCenter can be closed once a trade study is submitted, thus freeing up the computer for other uses. It eliminates the need to keep ModelCenter open throughout the trade study (important for long running Models).
- **Progress Monitoring:** Using Web browsers, users can check the status of runs and control their progress from anywhere.
- **Distributed Load Balancing:** CenterLink may maintain multiple groups of computers according to their types, ownership, and accessibility, and each group may maintain its own load balancing mechanism. This non-central way of load balancing improves performance and reliability of the system when large trade studies are performed at the same time.
- **Central Repository of Data:** As results are returned to CenterLink, the data are stored in the central database automatically. This prevents loss of valuable computation results and makes it easy to share the results among team members.
- **Reuse of Existing Computing Facilities:** It is possible for an organization to reuse existing load balancing tools on clusters through custom connectors that are provided from the CenterLink architecture.

V. Baseline Performance Test

Ideal speed up is the ultimate goal of load balancing systems. For example, if a job is distributed over 10 computing nodes, the ideal wall-clock time of job execution should be one tenth of the time required for a single node case. In practice, ideal speed up is not possible due to overhead in load balancing system or inherently serial portions of the computation.

This section presents a baseline test of CenterLink load balancing. There are two main sources of computational overhead in CenterLink. First, the load balancing service of CenterLink requires communication with computing nodes. This type of overhead is affected by the network bandwidth. Second, extra start-up time is

incurred because each computing node starts up a special version of ModelCenter (called NodeRunner) to run the integrated model in a batch mode. To measure such overhead, a dummy executable code was used that sleeps for a specified period of time without using any CPU time. The sleeper executable was run through CenterLink that distributed the runs over a cluster consisting of four Pentium IV Windows PCs.

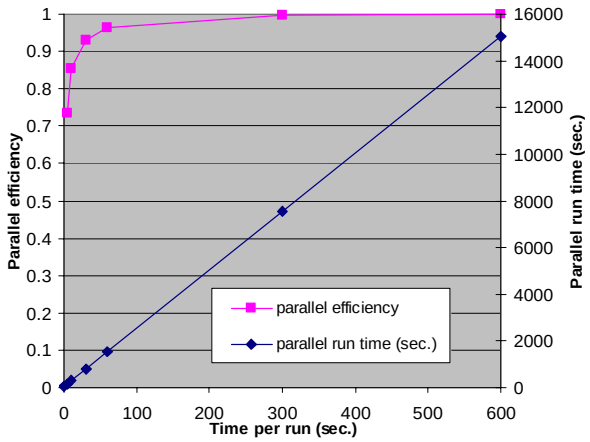

Figure 4: Baseline parallel efficiency of CenterLink on a size four cluster.

A DOE study of 100 runs was performed with various simulated run times ranging from 0 to 600 seconds. Each of the four nodes is expected to perform 25 runs on average. Figure 4 shows parallel run time and parallel efficiency with respect to the time per run. The zero run time case simulates a code running instantly. This case took 33 seconds when running through CenterLink, which can be attributed to inherent overhead of the load balancing system. The average overhead for each run over the 100 runs is estimated as 0.33 seconds. The parallel efficiency for this case is not close to the ideal efficiency of one because the overhead is not small compared to the run time. As the run time increases, the parallel efficiency improves. For example, the parallel efficiency was 74% when the run time was 5 seconds and it was 96% for a run time of 60 seconds. For the run time of 10 minutes, the speed up was 99.8%, close to the ideal efficiency.

VI. An Example of Accelerating Design Optimization

A. Problem Description: A Satellite Simulation

The parallel trade study capability was demonstrated for a space application. The numerical model includes the STK[7] satellite simulation software. In the satellite simulation scenario, there are two satellites that cover a region in Western Europe. STK numerically integrates the trajectory of the satellites and computes daily coverage hours of the target region. The optimization problem is to maximize the daily coverage hours subject to a cost constraint. Design variables include four satellite orbit parameters and a sensor parameter.

Objective:

Maximize Cov (daily coverage hours) (1.1)

Constraint:

$\text{Cost} \le \text{Cost}_{ub}$ (cost constraint) (1.2)

Variables:

$230 \le x_1 \le 280$ (Argument of Perigee of satellite #1, degree) (1.3)

$230 \le x_2 \le 280$ (Argument of Perigee of satellite #2, degree)

$150 \le x_3 \le 200$ (Right Ascension of Ascending Node of satellite #1, degree)

$150 \le x_4 \le 200$ (Right Ascension of Ascending Node of satellite #2, degree)

$200 \le x_5 \le 300$ (Focal length of sensor of both satellites, *m*)

The Darwin[8] genetic algorithm (GA) optimizer of ModelCenter was used to test load balancing performance of CenterLink. GA is well suited to CenterLink's coarse grained parallelism because it typically requires many function evaluations that can be computed concurrently. At each generation of a GA, Darwin sends a run table of the

population to CenterLink where the runs are distributed over the computing nodes. When the computation of each population is completed, Darwin generates the next generation and sends it to CenterLink.

The optimization model was created in ModelCenter by wrapping the STK scenario with the ModelCenter's STK plug-in. Wrapping technology defines inputs and outputs for the application code and automates its execution. The STK plug-in provides a graphical interface to easily wrap STK scenarios. Figure 5 shows the integrated model that includes the STK scenario, a cost model, and the Darwin GA optimizer.

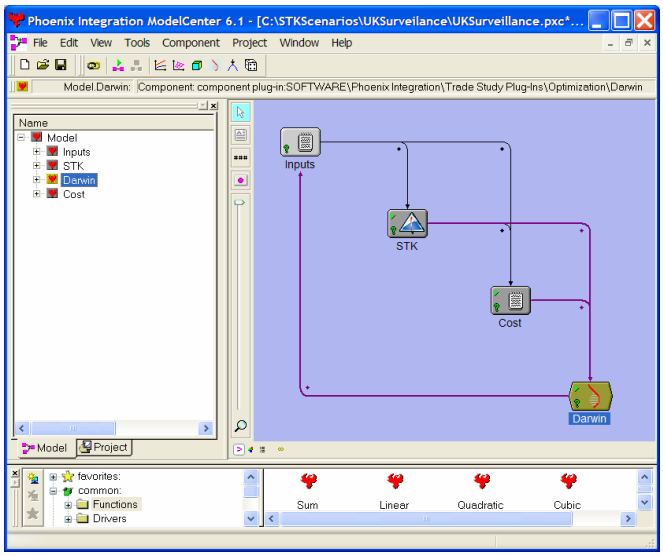

Figure 5: Satellite simulation model integrated using ModelCenter.

B. DOE Study

Before running optimization studies, parallel execution of the STK scenario was compared with serial execution for a DOE study. A three-level full factorial design with 243 (=3^5) runs was used for the five variable problem. When it was run using ModelCenter's serial execution, the DOE study took 8507 seconds to complete, averaging 33.2 seconds per run. Parallel execution using CenterLink took 2852 seconds on the four node cluster used in the baseline performance test. The speed up ratio was 2.98 and the corresponding parallel efficiency was about 75% (Figure 6). The parallel efficiency is lower than the 94% of the baseline performance results corresponding to the run time of 33 seconds as shown in Figure 4 because of additional overhead of running STK; the parallel execution mode of CenterLink needs to start up STK for each run while the serial execution mode uses the same STK session for all runs. The overhead of starting up STK was about 7~8 seconds per run, which was not negligible compared to the average run time of 33.2 seconds.

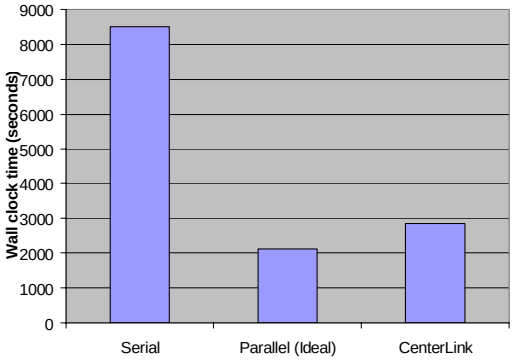

Figure 6: Load balancing performance for a DOE study. The ideal parallel run time is computed as the serial run time divided by four, the number of processors.

C. GA Optimization Study

A similar comparison was done for a GA run. The GA problem was set up to use a population size of 50 and to execute ten generations. Serial execution of the GA took 17078 seconds, while parallel execution on the four node cluster through CenterLink was 6048 seconds. The speed up ratio was 2.82 and the parallel efficiency was 71% (Figure 7). The parallel efficiency is 5% lower than the DOE case. The difference is attributed to the fact that the GA sent smaller sized run tables multiple times to CenterLink. For example, the GA submitted run tables of size 50 ten times, while the DOE run submitted a run table with 243 runs only once. Also, the STK startup overhead had non-negligible effects on the parallel efficiency.

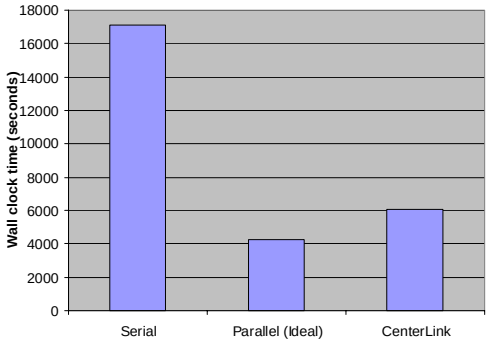

Figure 7: Load balancing performance for a GA optimization study. The ideal parallel run time is computed as the serial run time divided by four, the number of processors.

D. CAIV Study

The last test case was a CAIV (Cost As an Independent Variable) analysis[9]. Unlike the traditional design approach driven by system performance metrics, CAIV considers cost as an independent decision variable for acquisition process. An essential part of CAIV analysis is to provide trade-off information between performance and cost. One approach to obtain such trade-off information is to run a series of optimization cases with a varying cost constraint. In the STK example, $Cost_{ub}$ of Eq. 1.2 can be changed to generate optimization cases that maximize the coverage hours under varying cost limits. Twelve optimization cases were generated with different $Cost_{ub}$ values ranging from 1 to 12, in a scaled unit with respect to a base cost number. This CAIV scenario is another example that can leverage the coarse grained parallelism of CenterLink if the independent optimization cases are computed concurrently.

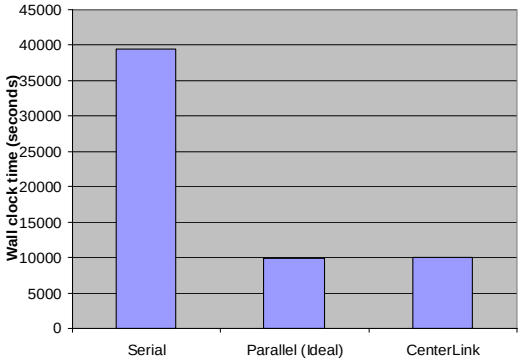

Figure 8: Load balancing performance for a CAIV study. The ideal parallel run time is computed as the serial run time divided by four, the number of processors.

For this example we ran the STK model in a low fidelity mode by adjusting the time step parameter of STK so that run time of the CAIV study was not excessive for the serial run. The Darwin GA was set up to perform 10 generations with a population size of 50. When the CAIV analysis was run through ModelCenter in the serial mode, the 12 optimization cases took 39438 seconds (Figure 8). The average run time of the GA was estimated as 3287 seconds. The same CAIV study was run through CenterLink using the four node cluster, and each of the nodes performed three GA runs. Note that this case differs from that described in Section C. In Section C, a single

optimization run was performed, and CenterLink was used to parallelize the individual STK analyses that made up that run. In this section, we are performing multiple optimization runs, and are using CenterLink to perform each of these optimization runs simultaneously. The individual analyses making up each run are performed in serial on each of the computing nodes.

The run time of CenterLink was 9988 seconds. Compared to the serial run, the speed up ratio was 3.95 and the parallel efficiency was 99%. In this case, the start-up overhead of STK is negligible compared to the GA run time. Therefore, the parallel efficiency is close to the ideal efficiency, as expected from the baseline performance testing results of Figure 4.

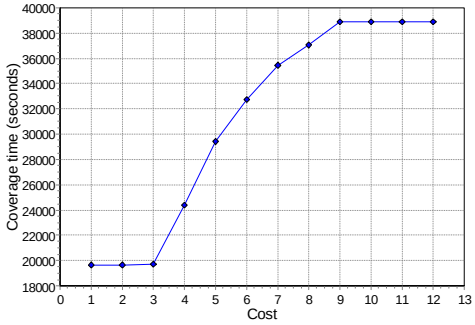

Figure 9: Cost versus system performance trade-off from a CAIV study.

Figure 9 shows the cost versus performance trade-off chart for the CAIV analysis. The trade-off clearly shows that the investment of more resources (i.e., cost) yields greater benefit in terms of improved performance. This is indicated by the steep slope of the curve when the cost level is around 4. However, the incremental gain tends to decline as the cost level increases and there is a certain limit beyond which performance of the system cannot be improved by committing more resources. This may mean that there are limitations that current technology cannot overcome. This information is very useful to make decisions in the early stages of the design process. CenterLink can help quickly generate this trade-off data by speeding up optimization runs.

VII. Deploying Standard Design Processes

CenterLink will serve as an infrastructure to deploy standard design processes over the Web. Design for Six Sigma (DFSS), design exploration techniques, and advanced MDO techniques are a few examples. In particular, the BLISS MDO methodology[1] is well suited to the coarse grained parallel capability of CenterLink. BLISS is an approach that allows design problems to be naturally decomposed into subsystem optimizations and system optimization.

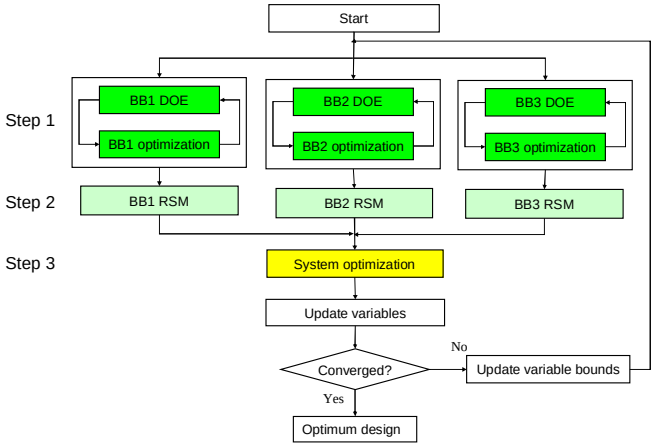

Figure 10: BLISS MDO workflow.

Figure 10 shows the workflow of BLISS MDO. Two levels of optimization of BLISS are performed in sequence. First, sub-systems are optimized with respect to local design variables specific to each disciplinary black box (BB) module. Each BB optimization is executed at a number of trial points set by a DOE technique in the BB input space (DOE-BB optimization loops in Figure 10). The optimization objective in each BB is a weighted sum of its output elements that couple it to other BBs. The weighting coefficients are set at the system level. Second, an approximate surrogate model, called response surface (RS), is generated for each element of the BB output (BB RSM in Figure 10), and RS from all the BB's are saved as the RS database. Third, a system optimization is performed with respect to system variables which include the design variables shared by multiple BB modules, the coupled response variables, and the weighting coefficients. The system optimization invokes the surrogate models and searches for the best system design objective that also satisfies the inter-BB coupling constraints. The system optimization objective function is a measure of merit for the entire system, e.g., the aircraft flight range. The system optimization draws all the data it needs from the surrogate model database. These three operations, BB optimizations, generation of approximations, and system-level optimization constitute one BLISS cycle which repeats iteratively until convergence or user's intervention.

The bulk of the numerical computation occurs in generating data for surrogate models. The key advantage of BLISS lies in the possibility to execute these steps autonomously and in parallel. A disciplinary or subsystem group (aligned with a BB in Figure 10) can carry out its BB optimization at all the DOE points concurrently to the extent limited only by the number of processors (DOE-level parallelism). Furthermore, multiple groups may carry out their DOE-BB optimizations simultaneously (BB-level parallelism). The Web centric architecture of CenterLink effectively supports the above two levels of parallelism in BLISS, and enables the system optimization by linking it to a pre-computed RS database.

VIII. Concluding Remarks

The purpose of this work is to develop capabilities to speed up trade studies for the design of complex systems. Combined with the ModelCenter process integration tool, CenterLink can execute the many analysis and optimization cases of a complex multidisciplinary analysis model in parallel. A Web-centric architecture was developed to provide trade study management services to distribute simulation cases to available computing resources over the network. The usability of parallel trade study clients is substantially improved because users can close the trade study client and monitor the progress via Web browsers. The transition from sequential runs to parallel trade studies is transparent to end-users. This capability allows engineers to focus on more design activities by cutting down efforts required to manage computing resources and address IT issues.

The CenterLink architecture is designed to achieve scalability in terms of speed-up and amount of trade study data being collected. The new load balancing capability was demonstrated for a satellite simulation application. The coarse grained parallelism was effectively utilized for a design of experiment study, a genetic optimization run, and a CAIV analysis, with their parallel computing efficiency ranging from 71% to 99%. CenterLink delivers the capability of using MDO to solve computationally expensive design problems of complex systems by exploiting potential of the concurrent computing technology.

Acknowledgements

Task performed by Phoenix Integration, Inc. was supported by a NASA contract, NNL04AA10C.

Disclaimer

Any mention of a commercial product or any report on such product in a NASA-sponsored activity does not constitute a NASA endorsement of such a product.

References

1. Sobieszczanski-Sobieski, J., Altus, T. D., Phillips, M., and Sandusky, R., "Bi-Level Integrated System Synthesis for Concurrent and Distributed Processing," *AIAA Journal*, Vol. 41, No. 10, 2003, pp. 1996-2002.

2. Phoenix Integration, Inc., “Improving the Engineering Process with Software Integration”, a technical white paper, 2002. (See also www.phoenix-int.com).

3. Phoenix Integration, Inc., “Accelerating Product Development through Grid Computing”, a technical white paper, 2004. (See also www.phoenix-int.com).

4. Woyak, S., Kim, H., Mullins, J., and Sobieszczanski-Sobieski, J., “A Web Centric Architecture for Deploying Multi-Disciplinary Engineering Design Processes,” AIAA-2004-4498.

5. Platform Computing Corporation, *Using Platform LSF on Windows*, 2004. (see also http://www.platform.com)

6. W3C, “Web Services Addressing 1.0 – SOAP Binding,” 2005. (see http://www.w3.org/TR/ws-addr-soap/)

7. AGI, “Discover Analysis & Visualization with STK”, (see www.agi.com).

8. Soremekun, G., Gürdal, Z., Kassapoglou, C., and Toni, D., “Stacking Sequence Blending of Multiple Composite Laminates Using Genetic Algorithms”, *Composite Structures*, Vol. 56, No. 1, 2002, pp. 53-62.

9. Criscimagna, N. H., “Cost As an Independent Variable (CAIV)”, *Journal of the Reliability Analysis Center*, 2nd Qt, 1998. (see http://rac.alionscience.com/pdf/2nd_Q1998.pdf)